Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2005)
K. Anjyo, P. Faloutsos (Editors)

Outside-In Anatomy Based Character Rigging

Michael Pratscher, Patrick Coleman, Joe Laszlo, and Karan Singh†

University of Toronto

Abstract

For believable character animation, skin deformation should communicate important deformation effects due to underlying muscle movement. Anatomical models that capture these effects are typically constructed from the inside out. *Internal tissue is modeled by hand and a surface skin is attached to, or generated from, the internal structure. This paper presents an* outside–in *approach to anatomical modeling, in which we generate musculature from a predefined structure, which we conform to an artist–sculpted skin surface. Motivated by interactive applications, we attach the musculature to an existing control skeleton and apply a novel geometric deformation model to deform the skin surface to capture important muscle motion effects. Musculoskeletal structure can be stored as a template and applied to new character models. We illustrate the methodology, as integrated into a commercial character animation system, with examples driven by both keyframe animation and recorded motion data.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Character modeling, Geometric deformation

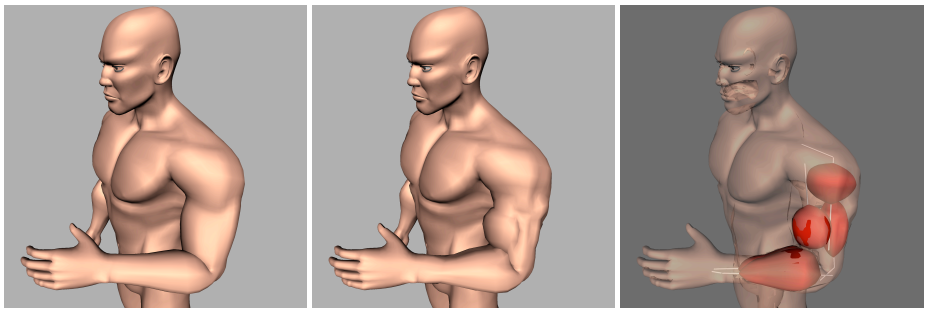
1. Introduction

In the process of character rigging, setting up models of skin deformation driven by the articulation of a control skeleton has become widespread practice [CHP89]. However, defining this deformation remains an extremely complex and time–consuming task. Animated characters for high–end film production require the explicit modeling of internal musculature to create believable deformation response. This musculature is typically constructed by hand to fit to character surface models carefully crafted by artists. Unfortunately, the expense of the approach, in terms of both computation and character construction, has precluded its adoption for most interactive applications. Instead, *skinning* algorithms are commonly used to define surface skin deformation as per–vertex weighted combination of control skeleton joint transformations. A neutral *rest pose* is used to initialize these per–vertex weight values. While the approach has many displeasing artifacts [LCF00] and can not capture effects of muscle deformation, it remains in widespread use due to its efficiency and the complete control it allows.

Given an internal anatomical model, the deformation of muscle, tissue, and skin can be computed through physical simulation to achieve detailed, realisitic motion [CZ92, TBHF03, TSB*04]. However, the computational expense and lack of direct control over the resulting skin deformation have precluded its adoption for character animation. Instead, geometric deformation approaches remain commonly used [SPCM97, WG97], as they allow for more direct animator control.

This paper presents a new *outside–in* approach to modeling musculature that, given a rest pose surface skin and control skeleton, constructs and adapts an anatomical structure to the given skin. The approach is general, in that it can be used with an arbitrary muscle deformation model. Change in muscle shape can be tied to change in skin shape using most existing approaches, including physical simulation or geometric deformation. As we are interested in simplifying the process of adding muscle–like deformations to characters for interactive applications, we illustrate the technique with a procedural model for muscle deformation [SPCM97] and a novel and powerful geometric model to map change in muscle shape to skin deformation.

† {map | patrick | jflaszlo | karan}@dgp.toronto.edu



(a) Smooth skinning. (b) Anatomic deformation. (c) Underlying musculature.

Figure 1: *Comparison of our interactive anatomic approach to smooth skinning alone, for a raised arm. In particular, note the accented bulge of the deforming bicep.*

1.1. Our Approach

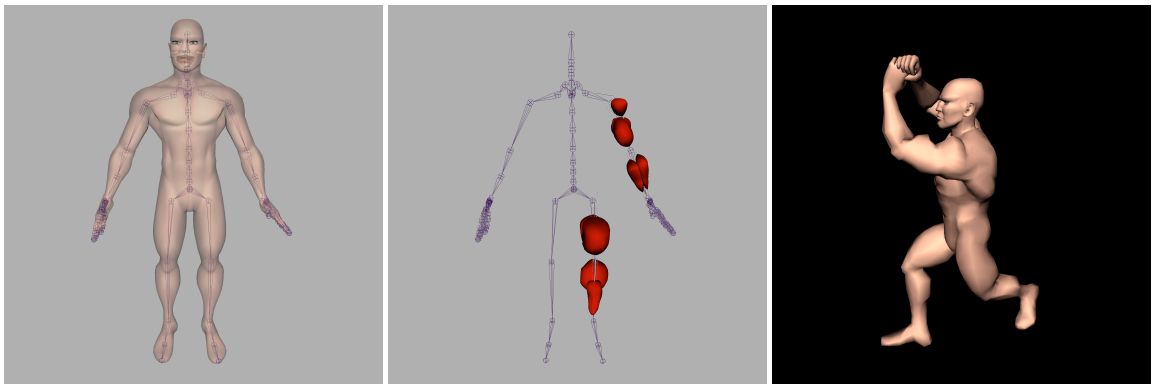
We begin with with an artist–sculpted character skin model and control skeleton (Figure 2a). The user can then fit any number of opposable muscle pairs to the bones of the control skeleton. Algorithmically, we do this by partitioning a mesh representation of the surface into regions, each corresponding to a bone using a number of heuristics (Section 3.1). For each region, the user specifies a partitioning plane for the muscle pair, which we use to fit two ellipsoidal muscles and connecting tendons (Section 3.2). The user can customize the connections and sizes of the muscles and tendons, and these parameters can be saved to a *musculoskeletal template* file for reapplication to new characters. Thus, an entire musculature can be quickly applied to appropriate new characters. After generating the muscles and tendons, the geometry of the muscles is reshaped to conform to the local surface of the character. The user can also edit the final shapes. Figure 2b shows the musculature generated to conform to the given character skin.

Given this generated musculature, we can apply any muscle deformation method, as well as any skin deformation technique that responds to changes in muscle shape. To address interactive applications and situations where an animator has direct control, we consider geometric deformation models. We consider two approaches for modeling change in muscle shape. First, if the original character has only been sculpted in a single pose, we use an extension of the volume preserving deformation technique described by Scheepers and colleagues [SPCM97] to change the muscle shape as the control skeleton is manipulated. Alternately, if multiple poses of the character have been sculpted, we generate the musculature for each pose and use sparse data interpolation to deform the muscle shape. Given either model of muscle deformation, we map the change in shape to the character skin using a new geometric deformation model that provides an animator with a number of intuitive controls to customize the final skin appearance (Section 4.2), as shown in Figure 2c.

Figure 1 compares our approach to a typical vertex–weighted transformation skinning solution (Figure 1a). The results of using our approach on the shoulder, upper arm, and lower arm are shown in Figure 1b. Figure 1c shows the underlying musculature deformed using our volume preservation technique. In particular, note the bulging of the bicep apparent in our solution, while the upper arm retains an unnatural rested appearance with the vertex–weighted skinning solution.

2. Related Work

Purely geometric approaches for deforming a character skin surface from skeletal articulation remain, by far, the most commonly used for interactive applications. In general, vertex deformations are constructed as a weighted blend of skeletal joint transformations (See [LCF00]). Vertex weights are determined by proximity to skeletal segments for a given rest pose. This approach produces smooth deformation, but the results appear closer to a flexed pipe than to deforming tissue. Indirect deformation models drive skin deformation using various functions, which are in turn driven by skeletal articulation [CHP89, MT97, SK00]. Typically, some variation of a free–form deformation lattice [SP86] is attached to the skeleton to define the deformation as a spatial embedding. Recently, example–based approaches have gained in popularity [LCF00, KJP02, WP02, ACP02, MG03]. The surface skin is sculpted in various poses, and deformation is defined as sparse data interpolation [ACP02, LCF00] or by fitting to parameters of an efficient model [MG03]. While these approaches are simple and efficient and provide direct animator control, they do not easily capture important nuances of realistic motion, especially effects of muscular deformation.



(a) Input skeleton and skin. (b) Fitted template musculature. (c) Anatomic deformation resulting from character articulation.

Figure 2: *Modeling of anatomic structure and the resulting effect on a posed character.*

Direct physically–based approaches tie the surface skin to the control skeleton, often with an internal volumetric mesh, and simulate dynamics to determine skin deformation [CGC*02, LTW95, JP02]. Dynamic models include damped mass–spring systems [LTW95], finite element models [GTT89, CGC*02], and modal analysis [JP02]. Physical simulation systems are not popular in commercial animation production, as they do not provide an animator with direct, intuitive, and keyframe–precise control over skin appearance. Hybrid approaches that overlay simulated secondary motion (e.g. hair, clothing) onto purely geometric character models have proven to be the most effective.

Anatomical models explicitly model internal structure to capture important effects such as muscle bulges that result from joint articulation. Early comprehensive systems [SPCM97, Wil97, WG97] geometrically deform muscle models, from which an external skin is generated. Schneider and Wilhelms [SW98] adapted their deformation approach to use artist–sculpted character surfaces, although anatomical structure had to be explicitly modeled to fit the surface. More recent systems [KHYS02, SWG02, KHS03, AHS03] conform a set internal musculature to provide animation controls. In contrast, we construct and adapt an arbitrary internal structure to a given character skin. Simulation models for muscle [CZ92, TBHF03, TSB*04] capture more detailed nuances of realistic large–scale muscle motion, although their computational complexity and lack of direct control have precluded their widespread adoption for animation.

3. Muscle Generation

To generate muscles for a given character skin and skeleton, we follow a two–stage process. First, we partition a mesh representation of the surface skin into regions corresponding to areas of muscle influence, and we create initial ellipsoidal muscles, which are attached to the control skeleton by tendons. Users can then refine the precise layout of the generated muscle primitives. Such customizations can be saved to a musculoskeletal template file for reapplication to new character skins. After the positions of the initial ellipsoidal muscles have been generated, we modify the muscle geometry to better conform to the artist–sculpted character skin.

3.1. Skin Segmentation

To fit ellipsoids to the character skin, we segment the surface mesh into regions, typically corresponding to expected muscle pairs, one region per pair. Muscles lie along bones, rather than joints; therefore, we segment based on proximity to the "bones" of the control skeleton, which geometrically correspond to line segments. For each vertex, we initially assign it to the region corresponding to the bone whose line segment has minimal distance.

Proximity segmentation alone has a number of limitations. First, the sparse, simplified nature of control skeletons results in many points being incorrectly assigned. For example, torso vertices of a character mesh in the common "da Vinci" pose are often assigned to arm regions. To compensate for this effect, we also require the vertex to pass a surface normal direction test to be assigned to a bone. In particular, we require that the surface normal point away from the vertex's closest point along the line of the bone. For a given vertex $\mathbf{p}$, surface normal $\mathbf{n}$, and projection onto the bone $\mathbf{p}'$, the acceptance test is $\mathbf{n} \cdot (\mathbf{p} - \mathbf{p}') \geq 0$. By incorporating this test, each vertex is assigned to the region corresponding to the closest bone, such that the corresponding surface normal points away from the bone.

Character surfaces with high–frequency geometry are also problematic. In particular, small details in regions of the nose and ears cause some surface normals to point inward, which causes inappropriate rejections. To alleviate this problem, rather than using the true surface normals, we use a low–frequency copy of the normals by applying Laplacian smoothing. For a given surface normal $\mathbf{n}$ and one–ring neighborhood N, the cached low–frequency normal $\mathbf{n}'$ is computed as follows:

$$\mathbf{n}' = (1-\lambda)\mathbf{n} + \lambda\left(\frac{\sum_{i\in N}\mathbf{n}_i}{||\sum_{i\in N}\mathbf{n}_i||}\right)$$

We iterate this smoothing pass a number of times to remove an appropriate level of detail. We provide the parameter λ to control the degree of normal smoothing. For the human–like models of the density we use (about 10,000 vertices), we have found 25 iterations with $\lambda = 1$ to work well. Users can optionally adjust these parameters to interactively obtain a good segmentation. Figure 4a illustrates a skin segmentation.

3.2. Ellipsoidal Muscle Fitting

After segmenting the mesh, we generate an elliptical muscle pair for each desired region. Users can create such pairs—or a single muscle–and then customize the muscle attachment. Parameters for such edits can also be read from a file to generate fully customized musculature rigs (described below). Each muscle's attachment is described by the tuple $\{a,i,i',o,o',\mathbf{p}_i,\mathbf{h}_{i'},\mathbf{p}_o,\mathbf{h}_{o'}\}$. a ... o' are indices into the transformations of the control skeleton, and $\mathbf{p}_i$ ... $\mathbf{h}_{o'}$ are points in the local space of the transformations with corresponding indices. We use four points to model muscle operation—the origin and insertion points $\mathbf{p}_o$ and $\mathbf{p}_i$, and intermediate hinge points $\mathbf{h}_{i'}$ and $\mathbf{h}_{o'}$. These hinge points allow tendons to be defined as piecewise linear curves to conceptually model the wrapping of tendons around bone without having to explicitly model skeletal shape. a is the index of the transformation whose bone segment corresponds to the region to which the muscle is fit. This model is illustrated in Figure 3.

Before fitting ellipsoid muscles, we optionally partition the region into two sub–regions to generate a muscle pair. The normal vector θ defines a partitioning plane through the center of the bone segment, which we use to partition the region. We have developed a tool that allows users to interactively specify this orientation by rotating a plane, as illustrated in Figure 4b.

For each region (or sub–region), we calculate the mean μ and covariance matrix Σ of the points. We perform singular value decomposition on Σ, and use the eigenvectors to align the axes of the ellipsoid. The square roots of the corresponding eigenvalues correspond to the half–lengths of the ellipsoid axes and thus map to the scaling components of the ellipsoid's transformation. Theoretically, this approach fits the ellipsoid well, as it corresponds to aligning a Gaussian distribution in space such that random samples would best explain the vertex locations. Figure 4c shows an example of best–fit ellipsoidal muscles.

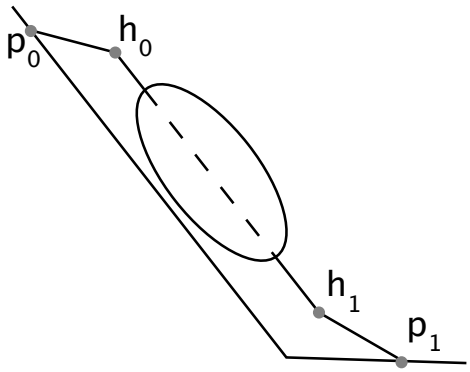


Figure 3: *Geometric structure of a generated muscle.*

3.3. Tendon Attachment and Muscle Shaping

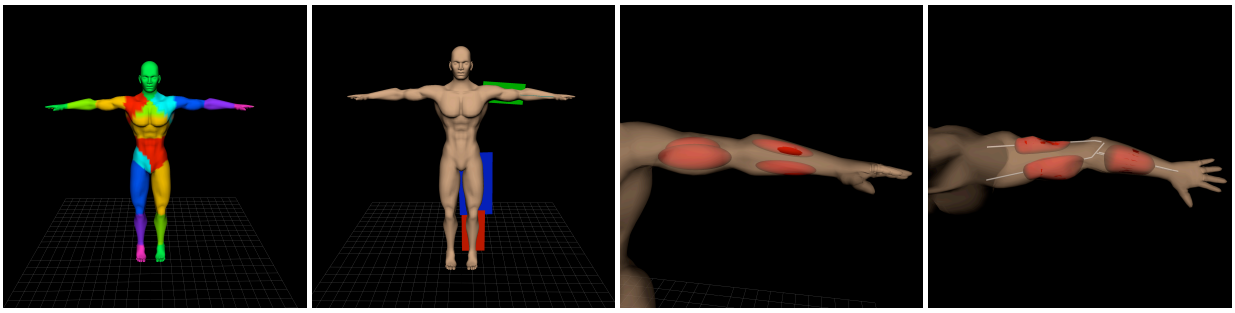
Before reshaping the muscle, we create insertion and origin points and corresponding hinge points. By default, the insertion and origin lie at the endpoints of the bone segment and the hinge points lie at the endpoints of the major axis. These can be interactively repositioned by the user or read from a template file. Fixed–length tendons are generated through the piecewise linear path from the origin/insertion, through the hinges, and connecting to the major axis endpoints.

Given the ellipsoidal muscles, we adapt their shape to conform to the character skin, allowing us to capture the deformation of underlying tissue. To do this, we inflate the ellipsoid toward, but not past, the skin surface, relative to the major axis. For an ellipsoid vertex $\mathbf{p}$ and corresponding projection to the major axis $\mathbf{p}'$, we find the intersection point $\mathbf{p}_i$ with the character skin, along the ray $\mathbf{p}-\mathbf{p}'$. To produce smooth muscle geometry, we exponentially attenuate this sculpting along the major axis. For a normalized distance d of $\mathbf{p}'$ from the major axis origin to the major axis endpoint, the final reshaped point $\mathbf{p}_r$ is as follows:

$$\mathbf{p}_r = \mathbf{p}' + \left(1 - e^{(d-1)\nu}\right)(\mathbf{p}_i - \mathbf{p}')$$

The parameter ν allows users to adjust the amount of attenuation.

When sculpting opposable muscle pairs, we additionally attenuate the sculpting about the radius of the major axis of the ellipsoid. If the ellipsoid–shaping ray points away from the partitioning plane surface, we make no adjustment. If it points toward the plane, we attenuate by $|(\mathbf{p}-\mathbf{p}')\cdot\theta|$. Figure 4d shows a reshaped muscle with tendon attachments.



(a) Surface segmentation. (b) Muscle pair partitioning planes. (c) Best–fit ellipsoidal muscles. (d) Conformed muscles.

Figure 4: *Muscle generation workflow. The character skin is first partitioned into regions corresponding to desired muscles (a). Each region is optionally partitioned to produce opposable muscle pairs by positioning partitioning planes (b). Ellipsoids are then fit to each region to create initial muscle geometry (c). The ellipsoidal muscles are then reshaped to better represent the internal tissue of the character skin, and tendons are created to attach the muscles to the bones (d).*

3.4. Musculoskeletal Templates

To assist the process of musculature creation, we allow users to store rigs with customizations in musculoskeletal template files. A template file contains parameters for a set of muscles and connection information for a labeled control skeleton. Users can save local customizations applied to a set of muscles or an entire musculoskeletal structure that can then be reapplied to new characters. Typically, such templates are authored once and reused for a set of characters with varying shape but similar body structure. For each muscle or muscle pair, we store values for all transformation indices of the muscle (as labels to skeletal transformations), as well as optional positions of the origin, insertion, and hinge points. Since we adapt muscle geometry to the shapes of individual character skins, templates define only the muscle arrangement and attachment, and we do not store explicit geometry. Figure 5 shows an example musculoskeletal template for a single muscle pair.

```
<boneMusclePair>
  <bone>
    <parentJoint>shoulder</parentJoint>
    <childJoint>elbow</childJoint>
  </bone>
  <muscle pairId="0" type="upper">
    <name>bicepsBrachii</name>
    <origin>
      <parentJoint>clavicle</parentJoint>
      <position type="absolute"
      x="-0.42" y="0.04" z="-1.82>
      </position>
    </origin>
    <oHinge>
      ...
    </oHinge>
  </muscle>
  <muscle pairId="0" type="lower">
    ...
  </muscle>
</boneMusclePair>
```

Figure 5: *Example musculoskeletal template file. One muscle pair is defined, and for each tendon attachment point, a space and position is given. Repetitive content is omitted for brevity.*

4. Deformation Models

Given a generated musculature, we can apply any model we wish to deform the muscles in response to the control skeleton—and in turn the skin, in response to the muscles. We choose to use geometric deformation models, as this provides direct animator control and allows the model to be used for interactive applications.

4.1. Muscle Deformation

We consider two approaches to deforming muscles in response to the control skeleton. First, we use a volume–preserving approach based on the work of Scheepers and colleagues [SPCM97]. Second, should an artist wish to sculpt a character skin in multiple poses, we allow them to generate or modify the musculature for any number of individual poses; we interpolate the generated muscles of each pose to deform the final muscle.

Procedural Deformation

We base our first muscle deformation model on ellipsoidal volume preservation. First, we constrain tendon length to

remain constant, regardless of attachment point configuration. To compensate for the change in overall length of muscles and tendons, we scale the muscle geometry such that tendon length is invariant. To create convincing muscle deformation, we preserve the volume of the underlying ellipsoid, and map the change in scale to the reshaped muscle. See [SPCM97] for details on volume preservation. We extend this deformation to take into account the state of fitness of the character. Users adjust a *tone* parameter, in the range $[0,1]$, to attenuate the change in scale. Thus, physically fit characters who would have high muscle tone have fully deforming muscle. Less fit characters will have lesser change in muscle shape, modeling the tendency of other intermediate tissue to not contract as muscle tissue does.

Pose-Based Deformation

As an alternative approach, we allow multiple sculpted poses and create corresponding muscles for each. We then combine the muscles of each pose into a shape interpolation model. Given the various sculpted poses, scattered data interpolation [LCF00] provides intermediate muscle shapes, based on new skeletal poses. For the case of a muscle attached to a single degree-of-freedom joint, we instead map the shapes to values of the single parameter, and interpolate between neighboring shapes to continuously deform the muscle.

4.2. Skin Deformation

To geometrically model skin deformation, we have designed a powerful ellipsoid–based deformation model that allows deforming and moving muscle shapes to drive the deformation of nearby skin geometry. All deformation is radial, relative to the center of the local space of the ellipse (in which it is a unit sphere at the origin). This is important, as it allows for efficient computation. Thus, all deformation occurs along the vector $\mathbf{E}^{-1}\mathbf{p}-\mathbf{0}$ for an ellipsoidal transformation matrix $\mathbf{E}$ and vertex $\mathbf{p}$. Deformed points are returned to world space after deformation in the ellipsoid's space.

After deforming points in local space, we return them to world space in one of three ways. First, we can return points to world space using the same transformation matrix $\mathbf{E}$. Second, we can smoothly transition between $\mathbf{E}$ and the transformation of an accessory copy of the ellipse, $\mathbf{E}_2$. We refer to this as the *stretch* option, as it allows the user to move the accessory ellipse to "pull" the surface around, relative to the original ellipse. In terms of muscle deformation, this allows us to deform the character skin relative to the ellipsoid's rest pose position. We also allow more complex stretching paths to be specified with a parametric curve; we refer to this as the *trajectory* option. Finally, we can accommodate arbitrary star–shaped deformation objects to allow for general muscle shapes.

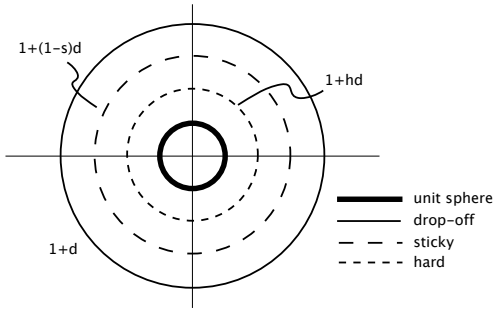


Figure 6: *Shells of the skin deformation model.*

4.2.1. Basic Deformation

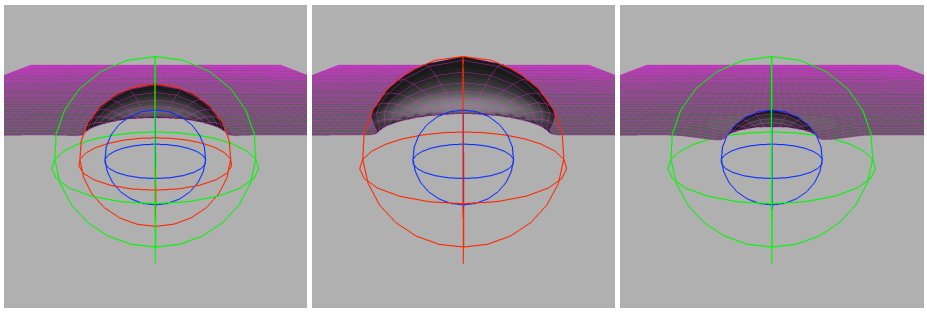
The deformation model consists of a number of shells in the local space of the deformer, as illustrated in Figure 6. First, a *sticky* shell determines a layer toward which all points deform. Second, a *hard* shell of radius $1+dh$ (outside the unit sphere) and an outer *drop–off* shell of radius $1+d$ define an attenuation region for the deformation. Points within the unit sphere undergo full deformation, and those outside the drop–off shell undergo no deformation. The user can adjust the shell radii through the more intuitive user parameters of drop–off (d), stickiness (s), and hardness (h) to achieve different qualities of deformation. We define the sticky shell radius relative to the drop-off shell as $1+(1-s)d$. We also provide an overall attenuation scalar w and a per–vertex attenuation scalar w_i.

For a given vertex $\mathbf{p}_l = \mathbf{E}^{-1}\mathbf{p}$ at radius r in local space, let $\mathbf{s}$ be its projection onto the sticky shell. Given a fall-off function $f(x)$ that varies smoothly from one to zero over the domain $[0,1]$, we define a piecewise continuous attenuation function $\beta(r)$. If $r > 1+d$ (beyond the drop–off shell), $\beta(r) = 0$. If $r < 1+hd$ (within the hard shell), then $\beta(r) = 1$. In the intermediate region, $\beta(r) = f(\frac{r-(1+hd)}{(1-h)d})$, resulting in a smooth attenuation. For the fall-off function, we require it to be tangent continuous; we use $f(x) = (1-x^2)^2$. Taking these terms into account, the local space deformation is as follows:

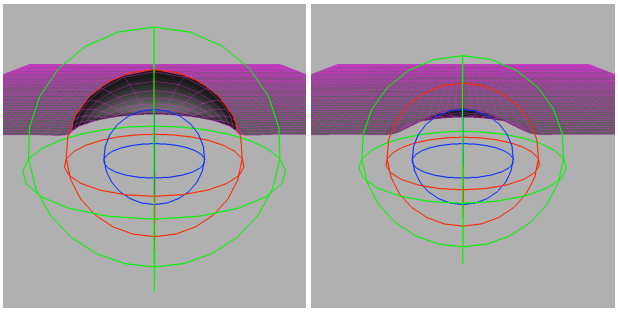
$$\mathbf{p}'_l = w w_i \beta(r) ||\mathbf{s}-\mathbf{p}_l|| \frac{\mathbf{p}_l}{||\mathbf{p}_l||}$$

For basic deformation, the final world space point is $\mathbf{p}' = \mathbf{E}\mathbf{p}_l$. For multiple deformations, we duplicate all parameters and apply shape changes in series.

Example edits of the deformation parameters are demonstrated in Figure 7. Given the rest state of the deformer, the surface deforms to lie on the sticky shell, in red (7a). Decreasing stickiness to zero expands the sticky shell to the drop–off shell, in green (7b). Increasing stickiness to one collapses the sticky shell down to the unit sphere, blue (7c). Increasing the drop-off parameter increases the size of the



(a) Rest state of the deformer. (b) Setting sticky to zero. (c) Setting sticky to one.



(d) Setting drop–off to 1.5 (e) Setting w to 0.5

Figure 7: *Modifying deformation parameters.*

attenuation region (7d). Reducing w attenuates all deformation(7e).

4.3. Stretch Deformation

To model stretching, we blend between the world space transformations of the rest position ellipse and the accessory ellipse. We transform the world space point to the local space of the rest ellipse and deform in the local space as described above. Rather than returning the point to world space through $\mathbf{E}$, we use an interpolation between $\mathbf{E}$ and $\mathbf{E}_2$. We define this interpolation as a piecewise continuous fall–off function $\alpha(r)$ from the unit sphere to the drop–off shell in local space. If $r < 1$, then points transform to world space via the accessory ellipsoid's transformation, and $\alpha(r) = 1$. If $r > 1 + d$, then points transform to world space via the rest position ellipsoid's transform, and $\alpha(r) = 0$. In the transition region, we use the same drop–off function as was used to attenuate the deformation, and $\alpha(r) = f(\frac{r-1}{d})$. The resulting transformation to world space is as follows:

$$\mathbf{p}' = \mathbf{E}\mathbf{p}_l + \alpha(r)^l(\mathbf{E}_2\mathbf{p}_l - \mathbf{E}\mathbf{p}_l)$$

The exponent l is presented to the user as a *slippage* parameter, which can be adjusted to empirically capture the appearance of flesh sliding over underlying tissue. We illustrate the use of this technique in Figure 8a.

4.3.1. Trajectory Deformation

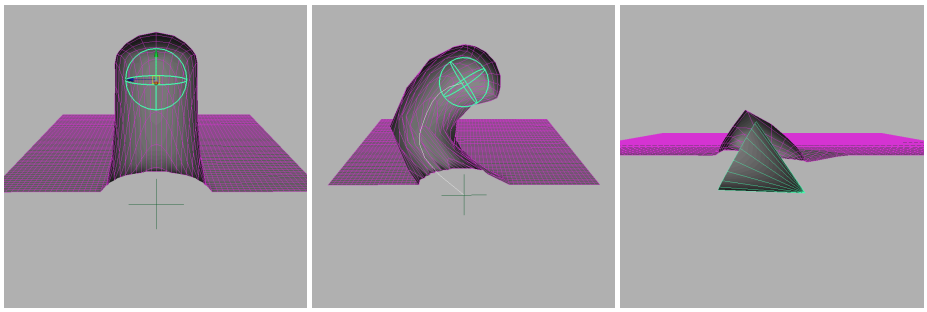
To generalize the stretch deformation to follow an arbitrary parametric curve between the ellipsoid centers, we find an intermediate point along the curve, as well as an intermediate vector from the center of the ellipsoids to the world space points. Without loss of generality, let the curve $\mathbf{c}(u)$ be parameterized such that $u \in [0,1]$, where $\mathbf{c}(0)$ corresponds to the origin of the rest position ellipsoid, and $\mathbf{c}(1)$ corresponds to the rest position of the accessory ellipsoid. The resulting world space transformation is now as follows:

$$\mathbf{p}' = \mathbf{c}(\alpha(r)^l) + \alpha(r)^l(\mathbf{E}_2\mathbf{p}_l - \mathbf{E}_2\mathbf{0}) + (1-\alpha(r)^l) * (\mathbf{E}\mathbf{p}_l - \mathbf{E}\mathbf{0})$$

Note that we interpolate vectors rather than points. Also, we clamp $\alpha(r)$ such that $\alpha(r) \in [0,1]$. The trajectory option is shown in Figure 8b.

4.3.2. Arbitrary Deformation Shapes

To generalize to arbitrary deformation shapes, such as our reshaped muscles, it is necessary to adapt the deformer to handle such shapes. We augment it to support star-shaped objects by precomputing per-vertex local space displacements



(a) Stretch deformation. (b) Trajectory deformation. (c) Deformation from a non-ellipsoid shape.

Figure 8: *Deformation options.*

from the unit sphere of the ellipsoidal deformer. Given an arbitrary shape, we first generate a best-fit ellipsoid as described in Section 3.2. We then compute the displacements to represent the given shape. Displacements are calculated from the furthest ray-shape intersections for a spherically parametrized sampling of rays, and these are interpolated over the surface of the sphere. When determining a skin point's deformation, we offset the deformation by the interpolated displacement, attenuated appropriately. Figure 8c demonstrates the use of a non–elliptical deforming object.

5. Results

We have tested our system with a number of examples. Figure 9 shows a comparison between our anatomic approach and smooth skinning. In Figure 9a, we show the results of smooth skinning, and in Figure 9b, we show our results. Figure 9c shows the underlying musculature. Figure 10 compares our approach for a number of other examples. The top row illustrates our technique, and smooth skinning alone is shown on the bottom. For each of these examples, note the effects of muscle bulging produced by our system, while smooth skinning alone produces an incorrect relaxed appearance. Refer to the accompanying video for further illustration.

6. Conclusion

We have presented a comprehensive system for automatically rigging and skinning characters to produce effective muscle deformation for interactive applications. Our approach to storing parameters of musculoskeletal structure in a template allows us to load and apply customized muscle systems to other artist–sculpted character skins. We have described an approach to constructing muscle geometry that conforms to the skin shape, to adapt to the sculpted musculature effects. Given a constructed muscle system, we provide two mechanisms that procedurally deform the muscles in a useful way. First, we incorporate volume preservation and a fitness model to procedurally deform muscle geometry using changes in skeletal structure. Second, we have described how multiple sculpted poses can be used to interpolate among muscle shapes as a skeleton changes.

In future work, we intend to investigate the extension of our system to more complex musculoskeletal models and geometric deformation approaches. The approach could be extended to allow arbitrary muscle shapes to be fit to a pre-existing character skin, and knowledge of bone geometry could be used to create more realistic muscle shapes. Finally, we plan to investigate methods by which physical simulation models might be adapted to conform to desired pose shapes.

7. Acknowledgements

Winnie Tsang co–developed a precursor system. DKP Effects provided one of the character models as well as motion data. John Hancock, Anand "Crispy" Agarawala, and Noah Lockwood assisted with video production. Michael Neff and Alexander Kolliopoulos provided feedback on an early draft, and Jack Wang provided logistic support.

References

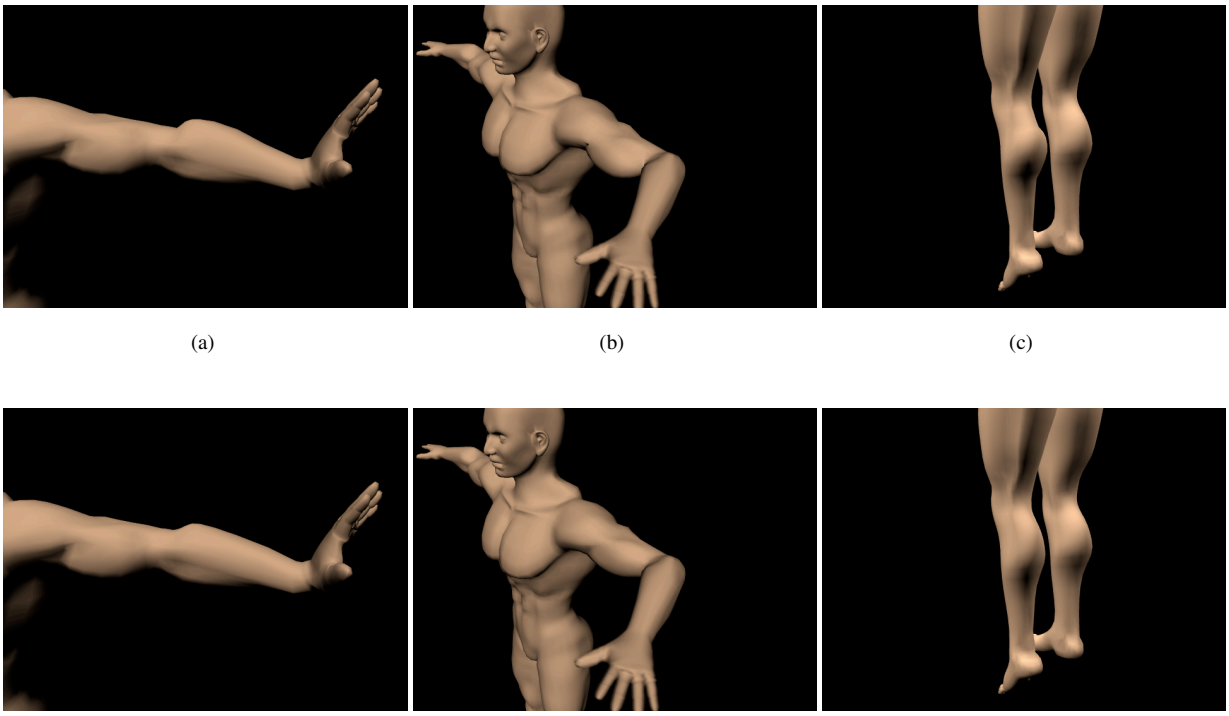
[ACP02] ALLEN B., CURLESS B., POPOVIĆ Z.: Articulated body deformation from range scan data. *ACM Trans. Graph. 21*, 3 (2002), 612–619.

[AHS03] ALBRECHT I., HABER J., SEIDEL H.-P.: Construction and animation of anatomically based human hand models. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation* (2003), pp. 98–109.

[CGC*02] CAPELL S., GREEN S., CURLESS B., DUCHAMP T., POPOVIĆ Z.: Interactive skeleton-driven dynamic deformations. In *SIGGRAPH '02: Proceedings*

(a) Smooth skinning alone. (b) Our anatomic approach. (c) Internal musculature.

Figure 9: *Results of applying our anatomic approach, in comparison to smooth skinning alone.*



(a) (b) (c)

(d) (e) (f)

Figure 10: *Our approach (top) in comparison to smooth skinning (bottom) for a number of examples.*

of the 29th annual conference on Computer graphics and interactive techniques (2002), pp. 586–593.

[CHP89] CHADWICK J. E., HAUMANN D. R., PARENT R. E.: Layered construction for deformable animated characters. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques* (1989), pp. 243–252.

[CZ92] CHEN D. T., ZELTZER D.: Pump it up: computer animation of a biomechanically based model of muscle using the finite element method. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques* (1992), pp. 89–98.

[GTT89] GOURRET J.-P., THALMANN N. M., THALMANN D.: Simulation of object and human skin forma-

tions in a grasping task. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques* (1989), pp. 21–30.

[JP02] JAMES D. L., PAI D. K.: Dyrt: dynamic response textures for real time deformation simulation with graphics hardware. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), pp. 582–585.

[KHS03] KÄHLER K., HABER J., SEIDEL H.-P.: Reanimating the dead: reconstruction of expressive faces from skull data. *ACM Trans. Graph. 22*, 3 (2003), 554–561.

[KHYS02] KÄHLER K., HABER J., YAMAUCHI H., SEIDEL H.-P.: Head shop: generating animated head models with anatomical structure. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2002), pp. 55–63.

[KJP02] KRY P. G., JAMES D. L., PAI D. K.: Eigenskin: real time large deformation character skinning in hardware. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2002), pp. 153–159.

[LCF00] LEWIS J. P., CORDNER M., FONG N.: Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), pp. 165–172.

[LTW95] LEE Y., TERZOPOULOS D., WALTERS K.: Realistic modeling for facial animation. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), pp. 55–62.

[MG03] MOHR A., GLEICHER M.: Building efficient, accurate character skins from examples. *ACM Trans. Graph. 22*, 3 (2003), 562–568.

[MT97] MOCCOZET L., THALMANN N. M.: Dirichlet free-form deformations and their application to hand simulation. In *CA '97: Proceedings of the Computer Animation* (Washington, DC, USA, 1997), IEEE Computer Society, p. 93.

[SK00] SINGH K., KOKKEVIS E.: Skinning characters using surface-oriented free-form deformations. In *Proceeding of Graphics Interface 2000* (2000).

[SP86] SEDERBERG T. W., PARRY S. R.: Free-form deformation of solid geometric models. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (1986), pp. 151–160.

[SPCM97] SCHEEPERS F., PARENT R. E., CARLSON W. E., MAY S. F.: Anatomy-based modeling of the human musculature. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), pp. 163–172.

[SW98] SCHNEIDER P., WILHELMS J.: Hybrid anatomically based modeling of animals. In *CA '98: Proceedings of the Computer Animation* (1998), p. 161.

[SWG02] SIMMONS M., WILHELMS J., GELDER A. V.: Model-based reconstruction for creature animation. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2002), pp. 139–146.

[TBHF03] TERAN J., BLEMKER S., HING V. N. T., FEDKIW R.: Finite volume methods for the simulation of skeletal muscle. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation* (2003), pp. 68–74.

[TSB*04] TERAN J., SIFAKIS E., BLEMKER S., NG THOW HING V., LAU C., FEDKIW R.: Creating and simulating skeletal muscle from the visible human data set. *IEEE Transactions on Visualization and Computer Graphics 11* (2004), 317–328.

[WG97] WILHELMS J., GELDER A. V.: Anatomically based modeling. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), pp. 173–180.

[Wil97] WILHELMS J.: Animals with anatomy. *IEEE Comput. Graph. Appl. 17*, 3 (1997), 22–30.

[WP02] WANG X. C., PHILLIPS C.: Multi-weight enveloping: least-squares approximation techniques for skin animation. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2002), pp. 129–138.